GETTING OUT OF A TAILORABILITY DILEMMA

Sebastian Draxler, Gunnar Stevens

Chair of Information Systems and New Media
University of Siegen
Hölderlinstraße 3
57068 Siegen
sebastian.draxler@uni-siegen.de
gunnar.stevens@uni-siegen.de

Abstract: This contribution shows the dilemma, which can occur if tailorability is applied to software artefacts. We introduce our thoughts on a method to overcome this dilemma and compare this new method to existing participation methods.

1 Introduction

End User Development is grounded in the fact, that a system cannot be fully specified at design time. There are several reasons for this fact: For example misunderstandings between users and developers while creating a new application or off-the shelf software which is used by an anonymous, heterogeneous user community. One method to reduce this drawback is tailorability. This idea commits developers to make different parts of an application tailorable, so users can adjust their behaviour to fit the work process. But this approach can lead into the following dilemma: To make a part tailorable, one has to identify this part in first place and has to know how it should be tailorable. A way out of the dilemma is to conceptualize tailorability not only as technical feature of a software artefact, but to see tailorability as a part of the evolutionary, participatory design process [FO02][WR95]. Grounded in the social-technical perspective we argue that End User Development can benefit from a technical infrastructure that support the identification of those critical parts and to get feedback, how users want to tailor these aspects. Therefore we have developed the concept of Participatory Design in Use (PaDU). In this workshop we want to discuss the background of the PaDU concept. [1]

2 Approaches for enhancing user participation

Participatory Design (PD) has always said that it is easier for the users to know their needs, when they have use experiences. Outside the concrete working practice it is a hard job for

[1] For technical details on an example implementation visit http://www.bscweasel.de.

the users to tell whether an application fit the work process or not. In particular this is true, if only specifications are available. Nevertheless PD has developed only a few methods that support users to express their needs using an application. Therefore we argue that PD has to develop post-deployment PD tools, which means these tools should be integral part of the delivered application. In the case of PaDU, the user should discover which parts of the application should be changed based on her use experience and within her use context. We want users to discover design flaws while doing their daily work. Furthermore a shared communication infrastructure between users and designers should be established to support geographical and spatial differences.

Traditional Participatory Design Prototyping is the traditional PD method to deal with the problem of the missing use experience. Prototyping can be used to create inexpensive working models of software artefacts. The original intention is to get early user feedback to reduce risks and costs. But prototyping doesn't support the requirement of using a working application nor is it in general possible to use it if temporal and spatial differences exist, because many details, which aren't implemented in the prototype, have to be communicated by developers or project leaders.

F/OSS Philosophy Free/Open Source Software philosophy grants the maximum freedom, anyone can change the software himself. In addition, F/OSS philosophy supports to change the working application based on the use experience. Following an "always beta" concept usually applications can be used in daily life at an early stage of the design process, but the design process is open to react on new insights. On the other hand, F/OSS is not aware of the different needs and skills of the different stakeholders. As a consequence, users have to deal with the same complexity as developers.

Usability Engineering With the concept of Remote Evaluation, [CHH98] described that users are able to create usability reports (so called critical incident reports [NMT03]) concerning problems in using software artefacts. Their basic idea was to make usability tests inexpensive [HC98]. To reach their goal, they used network technologies to overcome temporal and spatial differences between users and evaluators. They also demanded that users should create critical incident reports in real work situations.

3 Important classification criteria of participation methods

In the previous section we have seen, that several design schools can make a contribution to the issue of Participatory Design in Use. In this section we want to discuss a first draft of a classification scheme for PaDU like methods. The goal is to draw a landscape of PaDU and to see where similarities and differences between the different approaches are (Figure1).

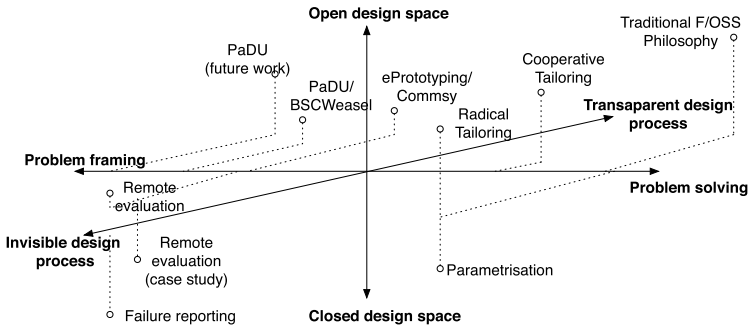

Figure 1: Classification of different approaches of post deployment user participation.

Structuring of the design activity Most participation methods address or support an explicit or implicit concept of design activity. As [Sch83] demonstrated, designing is more than just coding. He used the term of problem solving versus problem framing to describe the different design activities. Understanding and reflecting the use context is an important part of professional design. However, we believe this also applies to end users. Here we will use his terms to classify the different concepts structuring the users design activities.

Structuring of the design space Another distinction about participation methods is how they structure design space and the amount of room they leave for creativity. We use the term of open design space as a method that is aware of the user driven innovation, and the term closed design space if the method is primarily focused on anticipated issues.

Structuring of the design process The last classification dimension is motivated when comparing, for example, a common reporting tool of a commercial product to a participation tool for open source projects such as the bug reporting tool at Sourceforge. [NMT03] discovered that users, who create design ideas, often wish to receive feedback about the contribution. Also our informal talks with BSCWeasel/PaDU users indicate that the transparency of the design process have an effect on the motivation to participate. Therefore we want to distinguish between methods with transparent vs. invisible design processes.

4 Conclusion

The most concepts focus only on the aspect of problem solving. But our experience indicates that, for ordinary users support for problem framing is more important. A scenario of report creating users, as approved in remote evaluation comes close to what we think of, but we want users to reflect their problems, create a report and discuss possible solutions with the developers. Basically users should be able to develop design ideas for everything

they want to. To achieve this, an open design space seems to be the right choice. Thinking of PaDU as a reflection oriented method supports this requirement. As mentioned above, the transparency of the design process shouldn't be underestimated because it may motivate users to continue reporting their ideas. In addition, feedback from end users is usually not sufficient to understand context of usage and as such cannot and should not drive product iterations without being interpreted by experts and validated through contextual interviews. Therefore the transparent infrastructure should be designed as a communication infrastructure between the different stakeholders, in order to support the negotiation and clarification of design issues.

We introduced PaDU as a method to identify parts in software artefacts, which should be tailorable. We have identified several requirements for PaDU solution, such as a post deployment orientation, the need for remote participation, reflection orientation, an open design space and an open design process.

Build in tools like the PaDU realization in the BSCWeasel application and broadband internet access technologies enable users to send reports when ever they like, to attach illustrations and to update the application if requested functionality is available. Agile software engineering models like extreme programming will help to deliver working software artefacts even in early stages and to react very fast when a report leads to source code changes. Although PaDU is no silver bullet, our own design process already benefits from such a method. But in further research it will be interesting how PaDU can be applied in large software projects, how users can be better supported in reflecting about the application and their use context, how well PaDU can handle large user groups and how it can be applied to commercial applications.

References

[CHH98] José C Castillo, H. Rex Hartson, and Deborah Hix. Remote Usability Evaluation: Can Users Report Their Own Critical Incidents? *CHI98 Proceedings*, pages 253–254, 1998.

[FO02] Gerhard Fischer and Jonathan Ostwald. Seeding, Evolutionary Growth, and Reseeding: Enriching Participatory Design with Informed Participation. *PDC2002 Proceedings*, pages 292–298, 2002.

[HC98] H. Rex Hartson and José C. Castillo. Remote evaluation for post-deployment usability improvement. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces*, pages 22–29, New York, NY, USA, 1998. ACM Press.

[NMT03] David M Nichols, Dana McKay, and Michael B Twidale. Participatory Usability: supporting proactive users. pages 63–68, 2003.

[Sch83] Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books (June 1983), 1983.

[WR95] V. Wulf and M. Rohde. Towards an Integrated Organization and Technology Development. In *Symposium on Designing Interactive Systems (DIS'95)*, pages 55–64, Ann Arbor, MI, USA, 1995. ACM Press.